Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 20 (2014) 7 – 12

www.elsevier.com/locate/procedia

2nd International Conference on Ramp-Up Management 2014 (ICRM)

Greedy heuristics for distributed job shop problems

Ahmed Azab[a,*], B. Naderi[a,b]

[a] Department of Industrial and Manufacturing Systems, Universityo of Windsor, Windsor, Ontario, Canada
[b] Department of Industrial Engineering, University of Kharazmi, Karaj, Iran

* Corresponding author. Tel.: +1-519-253-3000; fax: +0-000-000-0000. *E-mail address:* azab@uwindsor.ca

Abstract

This paper studies the problem of scheduling distributed job shops where the classical single-facility job shop is extended to the multi-facility one. The problem is mathematically formulated by a mixed integer linear programming model. The small sized problems are optimally solved using commercial software of CPLEX. Three greedy heuristics, as well as adapting three well-known heuristics, are developed to solve large sized problems. The idea of the proposed heuristics is to iteratively insert operations (one at each iteration) into a sequence to build up a complete permutation of operations. The performance of the model and the six heuristics are comprehensive evaluated by numerical experiments. The results show the model and greedy heuristics are effective for the problem.

Selection and peer-review under responsibility of the International Editorial Committee of the "2nd International Conference on Ramp-Up Management" in the person of the Conference Chair Prof. Dr. Robert Schmitt

Keywords: Distributed job shop scheduling; mathematical model; greedy heuristics

1. Introduction

In the manufacturing environment, a new paradigm, called distributed manufacturing, is constituted. In distributed manufacturing, traditional single-facility entity has been changing to decentralized multi-facility ones. The advantages are low production costs, higher quality products, production flexibility and being closer to both suppliers and customers, so more responsive to market changes. In distributed job shops (DJS), we have a set of *f* identical facilities each of which consists of *m* machines. The problem of DJS becomes more sophisticated because two decisions have to be taken. The one is the allocation of jobs to facilities and the second is production scheduling of jobs. Moreover, we assume that no job crossing is allowed because it is very likely uneconomical or technologically difficult and impractical to transport a work-in-process job from one facility to anther to process its remaining operations. In distributed scheduling, makespan minimization is the minimization of maximum makespan among facilities.

Despite this trend, the literature of production scheduling focuses on single facility manufacturing. Jia et al. [1, 2] consider the distributed production scheduling problem and propose a genetic algorithm. Chan et al. [3] propose a genetic algorithm for distributed flexible manufacturing systems. Wang and Shen [4] write a book about distributed manufacturing where its focus is on planning and manufacturing problems rather than production scheduling. Regarding distributed flow shop, different metaheuristics are developed [5, 6, 7].

In this paper, the problem is mathematically formulated by a MILP model. Due to inherent hardness of DSJ, three well-known heuristics available in the literature of job shops (shortest processing time first (SPT), longest processing time first (LPT) and longest remaining processing time (LRPT)), are first adapted to DJS problem. To more effectively solve larger problems, three different greedy heuristics are also proposed. The algorithms are greedy because at each step, several alternatives are generated and the best one is selected. To evaluate the performance of the model and the six algorithms, different numerical experiments are conducted. The results show the model and greedy heuristics are effective for the problem.

The reminder of the paper is as follows. Section 2 describes the mathematical model. In Section 3, the adapted heuristics and proposed greedy algorithms are introduced.

Selection and peer-review under responsibility of the International Editorial Committee of the "2nd International Conference on Ramp-Up Management" in the person of the Conference Chair Prof. Dr. Robert Schmitt
doi:10.1016/j.procir.2014.05.025

Section 4 presents the numerical experiments. Finally, Section 5 concludes the paper.

2. Developed mathematical model

The problem of scheduling distributed job shops is mathematically modeled by a mixed integer linear program. The following parameters and indices are used in the developed model.

n	The number of jobs $j, k = \{0,1,2,\dots,n\}$
m	The number of machine $i, l = \{1,2,\dots,m\}$
f	The number of facilities $r = \{1,2,\dots,f\}$
$p_{j,i}$	The processing time of job j on machine i
$a_{j,i,l}$	1 if machine i is used immediately after machine l in the processing route of job j and 0 otherwise.
M	A large positive number

The model views the problem as positioning decisions. The decision variables are as such.

$X_{j,i,k}$	Binary variable taking value 1 if job j occupies k-th position of machine i and 0 otherwise.
$Y_{j,r}$	Binary variable taking value 1 if job j is assigned to factory r, and 0 otherwise.
$C_{j,i}$	Continuous variable for completion time of operation of the job j on machine i.

The MILP model is as follows.

$$Minimize\ C_{max} \tag{1}$$

$$Subject\ to:$$

$$\sum_{k=1}^{n} X_{j,i,k} = 1 \qquad \forall_{j,i} \tag{2}$$

$$\sum_{j=1}^{n} X_{j,i,k} = 1 \qquad \forall_{i,k} \tag{3}$$

$$\sum_{j=1}^{n} Y_{j,r} = 1 \qquad \forall_{j} \tag{4}$$

$$C_{j,i} \geq C_{h,i} + p_{j,i} - M\left(4 - X_{j,k,i} - \sum_{t=1}^{k-1} X_{h,i,t} - Y_{j,r} - Y_{h,r}\right) \qquad \forall_{j,h\neq j,k>1,i,r} \tag{5}$$

$$C_{j,i} \geq p_{j,i} \qquad \forall_{j,i} \tag{6}$$

$$C_{j,i} \geq C_{j,l} + p_{j,i} \qquad \forall_{j,i,l|a_{j,i,l}=1} \tag{7}$$

$$C_{max} \geq C_{j,i} \qquad \forall_{j,i} \tag{8}$$

$$C_{j,i} \geq 0 \qquad \forall_{j,i} \tag{9}$$

$$Y_{j,r} \in \{0,1\} \qquad \forall_{j,r} \tag{10}$$

$$X_{j,i,k} \in \{0,1\} \qquad \forall_{j,i,k} \tag{11}$$

Equation (1) is objective function. Constraint set (2) determines the position of each job on each machine. Constraint set (3) assures that each position on a machine is occupied by only one job. Constraint set (4) specifies the factory to which each job is assigned. Constraint set (5) is to show a machine can process at most one job at a time. Constraint set (6) is to show that the completion time of each operation is greater than its processing time. Constraint set (7) a job can be processed by at most one machine at a time. Constraint set (8) calculates the makespan. Constraint sets (9), (10) and (11) define the decision variables.

3. The heuristics

This section first adapts three well-known heuristics to DJS problem. To do this, a job-facility assignment rule is introduced. Using this rule, jobs are assigned to facilities, and then sequenced by the heuristic. Then, three greedy heuristics are proposed.

3.1. The three adapted rule-based heuristics (SPT, LPT and LRPT)

DJS includes two decisions: 1) job-facility assignment; 2) sequencing of jobs assigned to each facility. To apply well-known heuristics of job shops on DJS a job-facility assignment rule is required. The purpose is to partition jobs into facilities to smooth the workload in different facilities. As the measure of workload, one alternative is to consider the total processing time of each job j on different machines, i.e., $\sum_{i=1}^{m} p_{j,i}$.

However, even if we have facilities with the same total processing times, we may not obtain a good makespan. If jobs with large processing time on the very same machine are assigned to the same facility, makespan increases quite significantly. Therefore, this is not a proper strategy either. To better partition jobs to facilities, the workload on each machine is considered separately. In other words, jobs are assigned to the facility where the maximum workload of machines is minimized. Only considering this criterion is not still sufficient, since the processing route of jobs needs be considered. If jobs with the same processing route are assigned to the same facility, makespan increases in spite of the fact that the workload is balanced on each machine on its own.

To consider both factors (i.e., the processing times on each machine and processing routes), the following notation is used for workload. The workload of job j on machine i is as follows:

$$\text{workload(j,i)} = \left(\sum_{k\epsilon R_{j,i}} p_{j,k}\right) + p_{j,i} \quad \forall_{i,j}$$

where $R_{j,i}$ is the set of all machines preceding machine i in the processing of job j. In this case, machines in last stages of the processing route have more adjusted workload comparing

the machines in early stages. This notion is used in the adapted heuristics to make sure the workload is balanced at large for each facility.

Jobs of each facility are now sequenced using three well-known rules of SPT, LPT and LRPT. In SPT, for each machine there is a list of jobs sorted in non-descending order of processing times. Once a machine is available, the first job on its list is processed if the job is available and all of its preceding operations on rest of machines (according to job prescribed processing route) are already completed. If no such a job is available, the machine remains idle until such a job is available. The procedure of LPT is similar to SPT with a major difference. Unlike SPT, with the initial list corresponding to each machine, jobs are sorted in descending order of processing times. Note that schedules generated by heuristics are non-delay.

3.2. The proposed greedy heuristics

This section proposes three new high performing heuristics. They are construction heuristics revolving around the insertion neighborhood concept in a greedy fashion. The proposed heuristics are implemented using a permutation representation. A representation is the key for maintaining the search effectiveness.

To encode a solution of DJS problem, multiple operation-based permutations are used, one for each facility. The permutation of a facility expresses both jobs assigned to that facility and the relative order of the operations of those jobs on machines. As there are precedence constraints among the operations of each job, not all the permutations of the operations are feasible solutions. Thus, the classical operation-based permutation is not effective.

The following encoding scheme is developed. There are f operation permutations and a set of n_i operations corresponding to each job i. In this representation, each job index i appears a number of times matching to its number of operations. A job index appears only in one of the f permutations. By scanning the permutation from left to right, the kth appearance of a job index refers to the kth operation of the job. Hence, a permutation with repetition of job indices shows the order in which the operations of the job are processed. If a job number is repeated as the number of its operations, the solution is feasible.

3.2.1. Greedy Heuristic 1 (GH1)

In GH1, two decisions of job-facility assigning and operation sequencing are taken sequentially. Using the proposed job-facility assignment rule, the jobs are partitioned into different facilities. Then, operations of each facility are sequenced.

GH1 iteratively inserts operations, one at a time, into a sequence to build up a complete permutation of operations. The procedure can be described as follows: an initial random order of operations is built using its job indices where each job index appears as many as its operations (as discussed earlier). Then, job indices are one by one taken from the initial order and put into all the possible positions in sequence of scheduled job numbers. Finally, the best position is selected. The procedure repeats for the next job and so on and so forth.

Suppose a problem with f=2, n=7 and m=2. Imagine jobs 2, 3 and 6 are assigned to facility 1. The initial random order of 6 operations is $\{2,6,3,3,6,2\}$. At the first step, job 2 is taken. Since there is no scheduled operation, only one position exists. For the next job number, there are two positions. Therefore, we have two possible sequences: $\{2,6\}$ and $\{6,2\}$. The best sequence, among the possible ones, is selected. Assume that $\{2,6\}$ is the better one. To insert the next job index, there are three available positions and hence, three sequences

$$\{3,2,6\},\{2,3,6\} \text{ and } \{2,6,3\}.$$

Figure 1 shows the general outline of GH1.

```
Assign jobs to factories (job-factory assignment)
For k=1 to f do
    Determine random initial order of operations assigned to factory k
    For i=1 to q do   %q is the total number of operations assigned to factory k
        Take ith job number for the initial order
        For h=1 to i do
            Test inserting the job number into hth position
        Endfor
        Select the best position
    Endfor
Endfor
```

Figure 1. The general outline of GH1

3.2.2. Greedy Heuristic 2 (GH2)

This heuristic is also an insertion based heuristic. In GH1, two decisions of job-facility assignment and sequencing are sequentially taken (i.e., no interaction between the two decisions). Unlike GH1, job-facility assignment and sequencing are interactively determined in GH2. The jobs are initially sorted. At each step, one job is taken from the initial order, allocated to a facility using rule-based assignment; only then sequencing of its operations starts.

To initially sort the jobs, they are arranged in descending order of their total processing times across all machines (i.e., $P_j = \sum_{i=1}^{m} p_{j,i}$). The first f jobs are taken and each is assigned to a facility. For each facility, a permutation is built by its corresponding job index. Then, next job in the initial order, is taken. To assign it to a facility, the facility with the lowest makespan is selected.

To sequence its operations, the idea of iteratively inserting operations (one at each iteration) into a sequence to build up a complete permutation of operations has been utilized. The job index is added to the permutation of the selected facility for as many number as its operations. Each time, the job number is put into all the possible positions in sequence of scheduled job indices. Note that redundant sequences are discarded. Finally, the best position is selected. Figure 2 shows the general outline of GH2.

```
Arrange jobs in descending order of total processing times
 For j=1 to f do
     Take jth job in the initial order
     Assign the job to jth factory
Endfor
For k=f+1 to n do
     Select the factory k with the lowest makespan
     For i=1 to n_j do   %n_j is the number of operations of job j
          For h=1 to q_k + i do  %q_k is the total number of
                                 operations assigned to factory k
               Test inserting the job number into hth position
          Endfor
          Select the best position
     Endfor
Endfor
```

Figure 2. The general outline of GH2

3.2.3. Greedy Heuristic 3 (GH3)

Like GH2, this heuristic interactively takes two decisions of job-facility assignment and sequencing. The main difference is the job-facility selection rule. In GH2, each job is assigned to the facility with the lowest makespan. While with GH3, it performs in a greedier fashion and assign the job to the facility resulting in the lowest makespan after sequencing the operations of the job. That is, the assignments of jobs to all facilities are tested and the best one is selected. In this case, it is expected to end up with a better solution although more time-consuming. The other steps of GH3 are similar to GH2. The initial arrangement is in descending order of total processing times, etc.

4. Experimental evaluation

In this section, we evaluate the performance of the proposed mathematical model and six algorithms (SPT, LPT, TRPT, GH1, GH2 and GH3). To perform the evaluation, two experiments are conducted. In the first experiment, the model is evaluated in terms of both size and computational complexities. In the second experiment, the tested heuristics are compared with the optimal solution obtained by the model in the small instances. A set of larger instances is also used to compare the algorithms for the performance.

These algorithms are implemented in Borland C++ and run on a PC with 3.4 GHz Intel® Core™ i7-3770 CPU and 8 GB of RAM memory. The performance measure used in this research is percentage deviation index (PDI). It can be calculated as follows.

$$PDI = \frac{Alg - Min}{Max - Min} \times 100$$

where Alg is the makespan obtained by any of the algorithm. Min and Max are the lowest and largest makespans obtained for a given instance.

Two sets of instances are required to do the experiments mentioned above, one with small instances and the other with larger instances. An instance consist of the number of factories (f), the number of jobs (n), the number of machines (m) and the processing ($p_{j,i}$). Table 1 shows the considered level for each set. For the first set, when f=2, there are 8 combinations and when f=3, we only consider n=10 and m=3, 4 summing up to 8 combinations in total. For each combination, two instances are generated by random processing times taken from a uniform distribution between 1 and 99. For the second set, we use the instances of Taillard benchmark for job shops [8]. This benchmark includes 8 combinations for n and m, and 10 instances for each combination. It sums up to 80 instances. Each instance is solved by different levels of f; thus, there are 320 instances.

Table 1. The considered levels for the data generation

	The first set	The second set
f	{2, 3}	{2, 3, 4, 5}
n	{6, 8, 10}	{15, 20, 30,50,100}
m	{3, 4}	{15, 20}

4.1. Models evaluation

Two frequently used performance measures to evaluate MILP models are size and computational complexities Table 2 shows the size complexity of the model to formulate a problem with n jobs, m machines and f factories. Regarding binary variables, size complexity of the model is $O(n^2)$. Thus, it is quadratic in n. It is also linear in both m and f. Regarding constraints, the model is $O(n^3)$ (i.e., it is cubic in n). Considering m and f, the models are linear. The first conclusion is that the most influential parameter on the size complexity of the model is n. The other parameters m and f have only linear effect on the size complexity.

Table 2. The size complexity of the model

Factor (No. of)	Model
Binary variables	$n^2m + nf$
Continuous variables	nm
Constraints	$4nm + n(f+1) + nmf(n-1)^2$

The first set of data mentioned earlier is used to analyze the computational complexity of the model. Using CPLEX 12, the instances of this set are solved. The model is given a maximum time limit of 3000 seconds. Table 3 shows the results. The model solves 7 instances. The model solves all the instances up to 6 jobs. It rarely solves instances with 8 and 10 jobs.

4.2. Heuristics evaluation

Both data sets described earlier are used to evaluate the performance of the algorithms. These two data sets include small-sized instances solved by the proposed model and large-sized instances.

We first evaluate the general performance of the six tested algorithms versus the optimal solution of the first data set obtained by the models. Table 4 shows the results of these 7 instances. Among the algorithms, GH3 performs best by average optimality gap of 3.49%. Moreover, it results in the lowest makespan in 6 instances out of 7 ones.

Table 3. The results of the model

n	m	f	Model		
			Camx	Time(sec)	Optimality gap (%)
6	3	2	**25***	0.13	0%
6	3	2	**28***	14	0%
6	4	2	**32***	14.16	0%
6	4	2	**28***	4.8	0%
8	3	2	**28***	260	0%
8	3	2	29	3000	17%
8	4	2	37	3000	8%
8	4	2	40	3000	7%
10	3	2	33	3000	30%
10	3	2	33	3000	30%
10	4	2	28	3000	25%
10	4	2	42	3000	30%
10	3	3	**26***	10.05	0%
10	3	3	30	3000	20%
10	4	3	35	3000	2.80%
10	4	3	**33***	28	0%

Table 4. The results of the experiment with small instances

n	m	f	Op. C_{max}	Algorithms					
				GH1	GH2	GH3	LPT	SPT	LRPT
6	3	2	25	25	25	**25**	27	25	25
6	3	2	28	35	33	**31**	40	34	35
6	4	2	32	35	40	**32**	35	35	35
6	4	2	28	33	30	**29**	30	33	33
8	3	2	28	37	**28**	30	37	44	49
10	3	3	26	27	27	**26**	29	36	36
10	4	3	33	39	37	**34**	42	39	41
Average optimality gap				15.2	9.4	**3.4**	19.7	23.2	27.1

Table 5. The results of the algorithms on the large instances

n	m	Algorithms					
		GH1	GH2	GH3	LPT	SPT	LRPT
15	15	37.6	55.3	1.2	84.6	75.9	85.3
20	15	35.5	56.1	2.1	79.0	71.0	87.1
	20	38.5	60.2	1.1	84.2	82.2	87.2
30	15	48.6	62.6	0.1	81.1	61.4	89.3
	20	38.2	71.2	3.0	68.6	66.0	76.2
50	15	32.3	43.3	0.4	77.1	43.7	95.4
	20	40.1	70.2	2.8	66.6	44.6	90.9
100	20	31.5	54.9	2.1	82.9	32.7	91.6
Average		37.8	59.2	1.6	78.0	59.7	87.9

Table 6. The computational time of the tested algorithms (seconds)

n	Algorithms			
	GH1	GH2	GH3	LPT/SPT/LRPT
15	0.0048	0.0039	0.0128	0.0004
20	0.0153	0.0125	0.0414	0.0008
30	0.0391	0.0351	0.1297	0.0010
50	0.1607	0.1483	0.5879	0.0008
100	2.0580	1.9758	7.0359	0.0020

Now, we compare the tested algorithms on the second data set of larger instances. All its 320 instances are solved by the algorithms. Table 5 shows the results averaged by the combinations of (n, m). The proposed GH3 outperforms the other heuristics by far. It provides the average PDI of 1.21%. The second best heuristic is GH1 with average PDI of 37.83%. GH2 and SPT perform almost similarly. The worst performing heuristic is LRPT with average PDI of 87.93%. Note that the difference between GH2 and GH3 is in job-factory assignment and this shows how important job-factory assignment rule is.

We report the computational time elapsed by the algorithms. Table 6 shows the time averaged by the number of jobs. LPT, SPT and LRPT are simple rules, and their computational time is less than 0.01 second. Comparing the greedy heuristics, GH2 is faster than the two others.

5. Conclusion

Considering the trend of globalization, there is no research papers that directly study the distributed job shop problem. In this paper, a mathematical model was first developed for distributed job shops. The model was in form of mixed integer linear programs.

The models optimally solve small sized problems using CPLEX. Since the problem is an NP-hard one, three well-known heuristics were first adapted for the problem. Furthermore, three greedy heuristics were developed. The basic idea is to iteratively insert operations into a sequence to build up a complete permutation of operations in a constructive manner.

To evaluate the performance of the model and algorithms, two experiments were conducted. First, the model was evaluated based on both size and computational complexity. Then, the heuristics were evaluated against the optimal solutions obtained by the model and also compared against using a set of instances taken from Taillard benchmark.

References

[1] H.Z. Jia, J.Y.H. Fuh, A.Y.C. Nee, Y.F. Zhang, Web-based multi-functional scheduling system for a distributed manufacturing environment. Concurrent Engineering—Research and Applications 2002, 10(1), 27–39.

[2] H.Z. Jia, A.Y.C. Nee, J.Y.H. Fuh, Y.F. Zhang, A modified genetic algorithm for distributed scheduling problems. Journal of Intelligent Manufacturing 2003, 14(3–4), 351–362.

[3] F.T.S. Chan, S.H. Chung, L.Y. Chan, G. Finke, M.K. Tiwari. Solving distributed FMS scheduling problems subject to maintenance: genetic algorithms approach. Robotics and Computer-Integrated Manufacturing 2006; 22(5–6): 493–504.

[4] L. Wang, W. Shen. Process planning and scheduling for distributed manufacturing. London: Springer; 2007.

[5] J. Gao, R. Chen, W. Deng, An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem, International Journal of Production

Research 51(3), 2013, 641–651.

[6] S.W. Lin, K.C. Ying, C.Y. Huang, Minimising makespan in distributed permutation flowshops using a modified iterated greedy algorithm, International Journal of Production Research, 2013, http://dx.doi.org/10.1080/00207543.2013.790571.

[7] H. Liu, L. Gao, A discrete electromagnetism-like mechanism algorithm for solving distributed permutation flowshop scheduling problem, IEEE International Conference on Manufacturing Automation 2010, DOI 10.1109/ICMA.2010.17.

[8] E. Taillard, Benchmarks for basic scheduling problems, European Journal of Operational Research 64 (1993) 278–285.